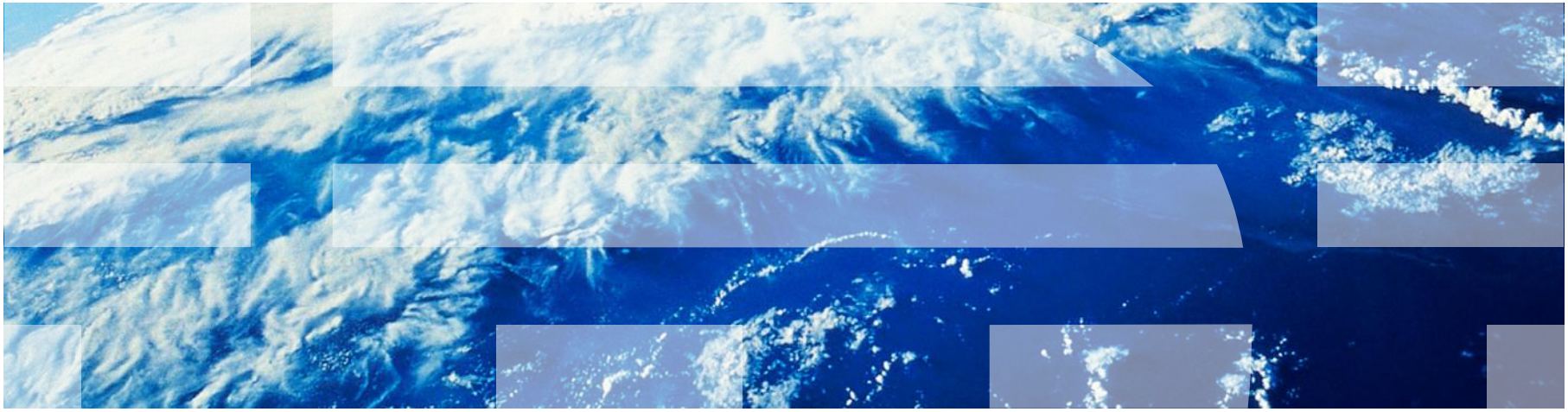
IBM Worklight Foundation V6.2.0 Getting Started

Using the LDAPLoginModule class to authenticate users with LDAP server in hybrid applications



20 June 2014

Trademarks

- IBM, the IBM logo, ibm.com, WebSphere, and Worklight are trademarks or registered trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.
- Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.
- Other company products or service names may be trademarks or service marks of others.
- This document may not be reproduced in whole or in part without the prior written permission of IBM.

About IBM

- See http://www.ibm.com/ibm/us/en/

Agenda

- LdapLoginModule overview
- Configuring the authenticationConfig.xml file
- Creating the client-side authentication components
- Examining the result

LdapLoginModule overview

- You can use the **LdapLoginModule** class to authenticate users with LDAP servers such as OpenLDAP or Active Directory.
- The **LdapLoginModule** class implements the **UserNamePasswordLoginModule** interface. Therefore, it must be used in conjunction with an authenticator that implements the **UsernamePasswordAuthenticator** interface. For example: **FormBasedAuthenticator**.
- For more information about how to implement **UsernamePasswordAuthenticator** interface, see module *Custom Authenticator and Login Module* in Table 9 of Tutorials and samples.
- In the following slides, you learn how to configure and use the **LdapLoginModule** class to protect various Worklight entities.

Agenda

- LdapLoginModule overview
- Configuring the authenticationConfig.xml file
- Creating the client-side authentication components
- Examining the result

Configuring the authenticationConfig.xml file (1 of 10)

- Add an authentication realm to the `<realms>` section of the **authenticationConfig.xml** file and call it **LDAPRealm.**

```
<realms>
    <realm loginModule="LDAPLoginModule" name="LDAPRealm">
        <className>com.worklight.core.auth.ext.FormBasedAuthenticator</className>
        <onLoginUrl>/console</onLoginUrl>
    </realm>
</realms>
```

- Use **FormBasedAuthenticator** in the `<className>` element because it implements the required **UsernamePasswordAuthenticator** interface.
- This realm uses **LDAPLoginModule** as a login module, which you define later.

Configuring the authenticationConfig.xml file (2 of 10)

- Add a login module to the `<loginModules>` section and call it **LDAPLoginModule**.

```
<loginModule name="LDAPLoginModule">
    <className>com.worklight.core.auth.ext.LdapLoginModule</className>
    <parameter name="ldapProviderUrl" value="ldap://10.0.1.2"/>
    <parameter name="ldapTimeoutMs" value="2000"/>
    <parameter name="ldapSecurityAuthentication" value="simple"/>
    <parameter name="validationType" value="searchPattern"/>
    <parameter name="ldapSecurityPrincipalPattern" value="{username}@myserver.com"/>
    <parameter name="ldapSearchFilterPattern" value="(&amp;(objectClass=user)(sAMAccountName={username})(memberof=
    <parameter name="ldapSearchBase" value="dc=myserver,dc=com"/>
</loginModule>
```

- Use **com.worklight.core.auth.ext.LdapLoginModule** in the `<className>` element.

Configuring the authenticationConfig.xml file (3 of 10)

- Add a login module to the `<loginModules>` section and call it **LDAPLoginModule**.

```
<loginModule name="LDAPLoginModule">
    <className>com.worklight.core.auth.ext.LdapLoginModule</className>
    <parameter name="ldapProviderUrl" value="ldap://10.0.1.2"/>
    <parameter name="ldapTimeoutMs" value="2000"/>
    <parameter name="ldapSecurityAuthentication" value="simple"/>
    <parameter name="validationType" value="searchPattern"/>
    <parameter name="ldapSecurityPrincipalPattern" value="{username}@myserver.com"/>
    <parameter name="ldapSearchFilterPattern" value="(&amp;(objectClass=user)(sAMAccountName={username})(memberof=
    <parameter name="ldapSearchBase" value="dc=myserver,dc=com"/>
</loginModule>
```

- The **ldapProviderUrl** parameter is mandatory. It defines the URL of your LDAP server.

Configuring the authenticationConfig.xml file (4 of 10)

- Add a login module to the `<loginModules>` section and call it **LDAPLoginModule**.

```
<loginModule name="LDAPLoginModule">
    <className>com.worklight.core.auth.ext.LdapLoginModule</className>
    <parameter name="ldapProviderUrl" value="ldap://10.0.1.2"/>
    <parameter name="ldapTimeoutMs" value="2000"/>
    <parameter name="ldapSecurityAuthentication" value="simple"/>
    <parameter name="validationType" value="searchPattern"/>
    <parameter name="ldapSecurityPrincipalPattern" value="{username}@myserver.com"/>
    <parameter name="ldapSearchFilterPattern" value="(&amp;(objectClass=user)(sAMAccountName={username})(memberof=
    <parameter name="ldapSearchBase" value="dc=myserver,dc=com"/>
</loginModule>
```

- The **ldapTimeoutMs** parameter is mandatory. It defines the timeout for LDAP server requests (in milliseconds).

Configuring the authenticationConfig.xml file (5 of 10)

- Add a login module to the `<loginModules>` section and call it **LDAPLoginModule**.

```
<loginModule name="LDAPLoginModule">
    <className>com.worklight.core.auth.ext.LdapLoginModule</className>
    <parameter name="ldapProviderUrl" value="ldap://10.0.1.2"/>
    <parameter name="ldapTimeoutMs" value="2000"/>
    <parameter name="ldapSecurityAuthentication" value="simple"/>
    <parameter name="validationType" value="searchPattern"/>
    <parameter name="ldapSecurityPrincipalPattern" value="{username}@myserver.com"/>
    <parameter name="ldapSearchFilterPattern" value="(&amp;(objectClass=user)(sAMAccountName={username})(memberof=
    <parameter name="ldapSearchBase" value="dc=myserver,dc=com"/>
</loginModule>
```

- The **ldapSecurityAuthentication** parameter is mandatory. It defines the type of authentication that is required by LDAP server. The usual value is `simple`, but you might need to contact LDAP administrator for a more appropriate value.

Configuring the authenticationConfig.xml file (6 of 10)

- Add a login module to the `<loginModules>` section and call it **LDAPLoginModule**.

```
<loginModule name="LDAPLoginModule">
    <className>com.worklight.core.auth.ext.LdapLoginModule</className>
    <parameter name="ldapProviderUrl" value="ldap://10.0.1.2"/>
    <parameter name="ldapTimeoutMs" value="2000"/>
    <parameter name="ldapSecurityAuthentication" value="simple"/>
    <parameter name="validationType" value="searchPattern"/>
    <parameter name="ldapSecurityPrincipalPattern" value="{username}@myserver.com"/>
    <parameter name="ldapSearchFilterPattern" value="(&amp;(objectClass=user)(sAMAccountName={username})(memberof=
    <parameter name="ldapSearchBase" value="dc=myserver,dc=com"/>
</loginModule>
```

- The **validationType** parameter is mandatory. It defines the type of validation that is performed. LdapLoginModule supports three types of validation: **exists**, **searchPattern,** and **custom**.

Configuring the authenticationConfig.xml file (7 of 10)

- The **validationType** parameter takes one of the following values:
  - **exists:** The login module tries to establish the LDAP binding by using the supplied credentials. The validation of the credentials is considered successful when binding is successfully established.
  - **searchPattern**: The login module first tries to run the **exists** validation. After such validation is successful, the login module issues a search query to the LDAP server context according to the **ldapSearchFilterPattern** and **ldapSearchBase** parameters. Credential validation is considered successful if a search query returns one or more entries.
  - **custom**: Use this value to enable a custom validation logic. The login module tries to run the **exists** validation After such validation is successful, the login module calls the `public boolean doCustomValidation(LdapContext ldapCtx, String username)` method. You can override this method by creating a custom Java™ class in your Worklight project and extending the `com.worklight.core.auth.ext.UserNamePasswordLoginModule` class**.**
    For more information about custom LDAP validation types, see the IBM® Worklight® Foundation user documentation.

Configuring the authenticationConfig.xml file (8 of 10)

- Add a login module to the `<loginModules>` section and call it **LDAPLoginModule**.

```
<loginModule name="LDAPLoginModule">
    <className>com.worklight.core.auth.ext.LdapLoginModule</className>
    <parameter name="ldapProviderUrl" value="ldap://10.0.1.2"/>
    <parameter name="ldapTimeoutMs" value="2000"/>
    <parameter name="ldapSecurityAuthentication" value="simple"/>
    <parameter name="validationType" value="searchPattern"/>
    <parameter name="ldapSecurityPrincipalPattern" value="{username}@myserver.com"/>
    <parameter name="ldapSearchFilterPattern" value="(&amp;(objectClass=user)(sAMAccountName={username})(memberof=
    <parameter name="ldapSearchBase" value="dc=myserver,dc=com"/>
</loginModule>
```

- The **ldapSecurityPrincipalPattern** parameter is mandatory. It defines the pattern in which LDAP security principal is sent to the LDAP server. You can use a `{username}` placeholder to inject the user name from the authenticator.

Configuring the authenticationConfig.xml file (9 of 10)

- Add a login module to the `<loginModules>` section and call it **LDAPLoginModule**.

```
<loginModule name="LDAPLoginModule">
    <className>com.worklight.core.auth.ext.LdapLoginModule</className>
    <parameter name="ldapProviderUrl" value="ldap://10.0.1.2"/>
    <parameter name="ldapTimeoutMs" value="2000"/>
    <parameter name="ldapSecurityAuthentication" value="simple"/>
    <parameter name="validationType" value="searchPattern"/>
    <parameter name="ldapSecurityPrincipalPattern" value="{username}@myserver.com"/>
    <parameter name="ldapSearchFilterPattern" value="(&amp;(objectClass=user)(sAMAccountName={username}))(memberof=
    <parameter name="ldapSearchBase" value="dc=myserver,dc=com"/>
</loginModule>
```

- The **ldapSearchFilterPattern** and **ldapSearchBase** parameters are optional. They apply only to the **searchPattern** validation type.

Configuring the authenticationConfig.xml file (10 of 10)

- Add a security test to the `<securityTests>` section of the **authenticationConfig.xml** file.
- You use this security test to protect the adapter procedure. Therefore, use the `<customSecurityTest>` element**.**

```
<customSecurityTest name="LDAPSecurityTest">
    <test isInternalUserID="true" realm="LDAPRealm"/>
</customSecurityTest>
```

- Remember the security test name because you use it in the following slides.

Agenda

- LdapLoginModule overview
- Configuring the authenticationConfig.xml file
- Creating the client-side authentication components
- Examining the result

Creating the client-side authentication components (1 of 14)

- Create a Worklight application.
- The application consists of two main `<div>` elements:
  - The **`<div id="AppBody">`** element is used to display the application content.
  - The **`<div id="AuthBody">`** element is used for authentication forms.
- When authentication is required, the application hides the `AppBody` element and shows the `AuthBody` element.
- When authentication is complete, it does the opposite.

Creating the client-side authentication components (2 of 14)

- Start by creating an AppBody.
- It has a basic structure and functions.

```
<div id="AppDiv">
    <div class="header">
        <h1>LDAPApp</h1>
    </div>
    <div class="wrapper">
        <input type="button" value="Call protected adapter proc" onclick="getSecretData()" />
        <input type="button" value="Logout"
               onclick="WL.Client.logout('LDAPRealm',{onSuccess: WL.Client.reloadApp})" />
    </div>
    <div id="resultDiv"></div>
</div>
```

- The buttons are used to call the **getSecretData** procedure and to log out.

Creating the client-side authentication components (3 of 14)

- The AuthBody element contains the following subelements:

```
<div id="AuthDiv" style="display:none">
    <div id="loginForm">
        Username:<br/>
        <input type="text" id="usernameInputField" value=""/><br />
        Password:<br/>
        <input type="password" id="passwordInputField" value=""/><br/>
        <input type="button" id="loginButton" value="Login" />
        <input type="button" id="cancelButton" value="Cancel" />
    </div>
</div>
```

  - A Username and a Password input fields.
  - A Login button and a Cancel button.
- The AuthBody is styled as `display:none`, because it must not be displayed before authentication is requested by server.

Creating the client-side authentication components (4 of 14)

- Finally, create a challenge handler.
- Use the following API to create this handler and implement its functionality.

```
var myChallengeHandler = WL.Client.createChallengeHandler("realm-name");

myChallengeHandler.isCustomResponse = function (response){
    return false;
};

myChallengeHandler.handleChallenge = function (response){
};
```

Use the **WL.Client.createChallengeHandler** method to create a challenge handler object. Supply a realm name as a parameter.

Creating the client-side authentication components (5 of 14)

- Finally, create a challenge handler.
- Use the following API to create this handler and implement its functionality.

```
var myChallengeHandler = WL.Client.createChallengeHandler("realm-name");

myChallengeHandler.isCustomResponse = function (response){
    return false;
};

myChallengeHandler.handleChallenge = function (response){
};
```

The **isCustomResponse** function of the challenge handler is called each time a response is received from the server. That function is used to detect whether the response contains data that are related to this challenge handler. It must return **true** or **false.**

Creating the client-side authentication components (6 of 14)

- Finally, create a challenge handler.
- Use the following API to create this handler and implement its functionality.

```
var myChallengeHandler = WL.Client.createChallengeHandler("realm-name");

myChallengeHandler.isCustomResponse = function (response){
    return false;
};

myChallengeHandler.handleChallenge = function (response){
};
```

If the **isCustomResponse** method returns **true**, the framework calls the **handleChallenge** function. This function is used to perform required actions, such as hide the application screen or show the login screen.

Creating the client-side authentication components (7 of 14)

- In addition to the methods that the developer must implement, the challenge handler contains functionality that the developer might want to use:
  - The `submitLoginForm` function sends collected credentials to a specific URL. The developer can also specify request parameters, headers, and callbacks.
  - The `submitSuccess` function notifies the Worklight framework that the authentication process completed successfully. The Worklight framework then automatically issues the original request that triggered authentication.
  - The `submitFailure` function notifies the Worklight framework that the authentication process completed with failure. The Worklight framework then disposes of the original request that triggered the authentication.

  **** Note: You must attach each of these functions to its object. For example:*** *`myChallengeHandler.submitSuccess()`*

- You use these functions during the implementation of the challenge handler in the next slides.

Creating the client-side authentication components (8 of 14)

- Create a challenge handler.

```
var LDAPRealmChallengeHandler = WL.Client.createChallengeHandler("LDAPRealm");

LDAPRealmChallengeHandler.isCustomResponse = function(response) {
    if (!response || !response.responseText) {
        return false;
    }
    var idx = response.responseText.indexOf("j_security_check");

    if (idx >= 0){
        return true;
    }
    return false;
};

LDAPRealmChallengeHandler.handleChallenge = f
        $('#AppDiv').hide();
        $('#AuthDiv').show();
        $('#passwordInputField').val('');
};
```

The default login form that is returned from the Worklight server contains the `j_security_check` string. If the challenge handler detects it in the response, it returns **true**.

Creating the client-side authentication components (9 of 14)

- Create a challenge handler.

```
var LDAPRealmChallengeHandler = WL.Client.createChallengeHandler("LDAPRealm");

LDAPRealmChallengeHandler.isCustomResponse = fun
    if (!response || !response.responseText) {
        return false;
    }
    var idx = response.responseText.indexOf("j_s

    if (idx >= 0){
        return true;
    }
    return false;
};

LDAPRealmChallengeHandler.handleChallenge = function(response){
        $('#AppDiv').hide();
        $('#AuthDiv').show();
        $('#passwordInputField').val('');
};
```

After the client application detects that the server sent a login form, which means that the server is requesting authentication, the application hides the AppBody, shows the AuthBody, and cleans up the **passwordInputField**.

Creating the client-side authentication components (10 of 14)

- Create a challenge handler.

```
$('#loginButton').bind('click', function () {
    var reqURL = '/j_security_check';
    var options = {};
    options.parameters = {
            j_username : $('#usernameInputField').val(),
            j_password : $('#passwordInputField').val()
    };
    options.headers = {};
    LDAPRealmChallengeHandler.submitLoginForm(reqURL, options,
            LDAPRealmChallengeHandler.submitLoginFormCallback);
});

$('#cancelButton').bind('click', function (
    $('#AppDiv').show();
    $('#AuthDiv').hide();
    LDAPRealmChallengeHandler.submitFailure
});
```

A click on a **login** button triggers a function that collects the user name and password from the HTML input fields and submits them to the server.
It is possible to set request headers here, and to specify callbacks.

Creating the client-side authentication components (11 of 14)

- Create a challenge handler.

```
$('#loginButton').bind('click', function () {
    var reqURL = '/j_security_check';
    var options = {};
    options.parameters = {
            j_username : $('#usernameInputField').val(),
            j_password : $('#passwordInputField').val()
    };
    options.headers = {};
    LDAPRealmChallengeHandler.submitLoginForm(reqURL, options,
            LDAPRealmChallengeHandler.submitLoginFormCallback);
});

$('#cancelButton').bind('click', function (
    $('#AppDiv').show();
    $('#AuthDiv').hide();
    LDAPRealmChallengeHandler.submitFailure
});
```

The form-based authenticator uses the hardcoded `j_security_check` URL component. You cannot have more than one instance of it.

Creating the client-side authentication components (12 of 14)

- Create a challenge handler.

```
$('#loginButton').bind('click', function (
    var reqURL = '/j_security_check';
    var options = {};
    options.parameters = {
            j_username : $('#usernameInput
            j_password : $('#passwordInput
    };
    options.headers = {};
    LDAPRealmChallengeHandler.submitLoginF
            LDAPRealmChallengeHandler.submitLogin
});

$('#cancelButton').bind('click', function () {
    $('#AppDiv').show();
    $('#AuthDiv').hide();
    LDAPRealmChallengeHandler.submitFailure();
});
```

A click on a **cancel** button hides the authBody, shows the appBody, and notifies the Worklight framework that authentication failed.

Creating the client-side authentication components (13 of 14)

- Create a challenge handler.

```
LDAPRealmChallengeHandler.submitLoginFormCallback = function(response) {
    var isLoginFormResponse = LDAPRealmChallengeHandler.isCustomResponse(response);
    if (isLoginFormResponse){
        LDAPRealmChallengeHandler.handleChallenge(response);
    } else {
        $('#AppDiv').show();
        $('#AuthDiv').hide();
        LDAPRealmChallengeHandler.submitSuccess();
    }
};
```

The callback function checks the response for the containing server challenge again. If a challenge is found, the `handleChallenge` function is called again.

Creating the client-side authentication components (14 of 14)

- Create a challenge handler.

```
LDAPRealmChallengeHandler.submitLoginFormCallback = function(response) {
    var isLoginFormResponse = LDAPRealmChallengeHandler.isCustomResponse(response);
    if (isLoginFormResponse){
        LDAPRealmChallengeHandler.handleChallenge(response);
    } else {
        $('#AppDiv').show();
        $('#AuthDiv').hide();
        LDAPRealmChallengeHandler.submitSuccess();
    }
};
```

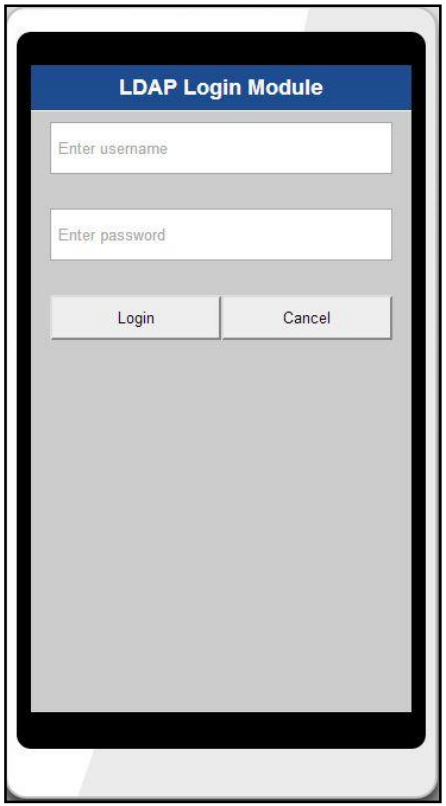
No challenge present in the server response means that authentication completed successfully. In this case, AppBody is shown, AuthBody is hidden, and the Worklight framework is notified about the authentication success.

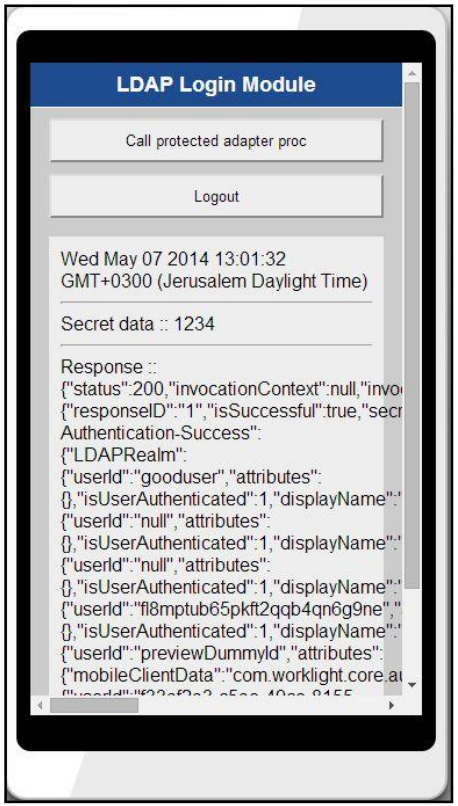
Agenda

- LdapLoginModule overview
- Configuring the authenticationConfig.xml file
- Creating the client-side authentication components
- Examining the result

Examining the result

- You can find the sample for this training module in the Getting Started page of the IBM Worklight Foundation documentation website at http://www.ibm.com/mobile-docs.





Notices

- Permission for the use of these publications is granted subject to these terms and conditions.
- This information was developed for products and services offered in the U.S.A.
- IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.
- IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:
  - IBM Director of Licensing
    IBM Corporation
    North Castle Drive
    Armonk, NY 10504-1785
    U.S.A.
- For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:
  - Intellectual Property Licensing
    Legal and Intellectual Property Law
    IBM Japan Ltd.
    1623-14, Shimotsuruma, Yamato-shi
    Kanagawa 242-8502 Japan
- **The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:** INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.
- This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.
- Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.
- IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.
- Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:
  - IBM Corporation
    Dept F6, Bldg 1
    294 Route 100
    Somers NY 10589-3216
    USA
- Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.
- The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.
- Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

COPYRIGHT LICENSE:

- This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.
- Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:
  - © (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. _enter the year or years_. All rights reserved.

Privacy Policy Considerations

- IBM Software products, including software as a service solutions, (“Software Offerings”) may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering’s use of cookies is set forth below.
- Depending upon the configurations deployed, this Software Offering may use session cookies that collect session information (generated by the application server). These cookies contain no personally identifiable information and are required for session management. Additionally, persistent cookies may be randomly generated to recognize and manage anonymous users. These cookies also contain no personally identifiable information and are required.
- If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent. For more information about the use of various technologies, including cookies, for these purposes, see IBM’s Privacy Policy at http://www.ibm.com/privacy and IBM’s Online Privacy Statement at http://www.ibm.com/privacy/details the sections entitled “Cookies, Web Beacons and Other Technologies” and the “IBM Software Products and Software-as-a-Service Privacy Statement” at http://www.ibm.com/software/info/product-privacy.

Support and comments

- For the entire IBM Worklight documentation set, training material and online forums where you can post questions, see the IBM website at:
  - http://www.ibm.com/mobile-docs
- **Support**
  - Software Subscription and Support (also referred to as Software Maintenance) is included with licenses purchased through Passport Advantage and Passport Advantage Express. For additional information about the International Passport Advantage Agreement and the IBM International Passport Advantage Express Agreement, visit the Passport Advantage website at:
    - http://www.ibm.com/software/passportadvantage
  - If you have a Software Subscription and Support in effect, IBM provides you assistance for your routine, short duration installation and usage (how-to) questions, and code-related questions. For additional details, consult your IBM Software Support Handbook at:
    - http://www.ibm.com/support/handbook
- **Comments**
  - We appreciate your comments about this publication. Please comment on specific errors or omissions, accuracy, organization, subject matter, or completeness of this document. The comments you send should pertain to only the information in this manual or product and the way in which the information is presented.
  - For technical questions and information about products and prices, please contact your IBM branch office, your IBM business partner, or your authorized remarketer.
  - When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you. IBM or any other organizations will only use the personal information that you supply to contact you about the issues that you state.
  - Thank you for your support.
  - Submit your comments in the IBM Worklight Developer Edition support community at:
    - https://www.ibm.com/developerworks/mobile/worklight/connect.html
  - If you would like a response from IBM, please provide the following information:
    - Name
    - Address
    - Company or Organization
    - Phone No.
    - Email address

Thank You